Test Platform for the Performance Evaluation of OPC-UA Servers for Fast Data Transfer Between Intelligent Equipment

Flavio González Vázquez
Ultraclean Technology and Micromanufacturing
Fraunhofer Institute for Manufacturing Engineering and Automation (IPA)
Stuttgart, Germany
e-mail: Flavio.Gonzalez@ipa.fraunhofer.de

***Abstract*— The ubiquity and widespread support of the Object Linking and Embedding for Process Control (OPC) Unified Architecture protocol in the automation industry and also outside it enable millions of intelligent manufacturing devices to effectively communicate and exchange information between each other in a secure and standardized manner. On the other hand, the ever growing need for large data transfers for predictive maintenance, process visualization and Internet of Things, among others, requires a precise knowledge of the protocol and system limitations in order to plan migrations or new installations. In this paper, a flexible test platform for the performance evaluation of OPC-UA systems is presented, along with preliminary findings and comparative performance measures of three different categories of PC-based OPC-UA systems.**

Keywords*— *Object Linking and Embedding for Process Control Unified Architecture (OPC-UA); performance evaluation; intelligent manufacturing systems; test platform; industrial automation.

I. INTRODUCTION

The requirements of quick and cost-effective integration and data visualization of heterogeneous intelligent automation systems has rapidly popularized the usage of OPC Unified Architecture (OPC-UA) as communication protocol in the automation industry. In comparison with the classic Object Linking and Embedding for Process Control (OPC) protocol that has been in use since 1996 for Windows-based systems, OPC-UA achieves platform independence, better scalability and a more secure approach based on newer standards, among other benefits. However, with now the majority of PLC and equipment manufacturers supporting the OPC-UA protocol, and given the fact that the protocol is based on the Transmission Control Protocol (TCP) that does not have real-time requirements, questions about the performance of the protocol and specially of different OPC-UA server implementations arise when planning new installations or considering upgrading existing ones. Different applications like inter-equipment data transfer for processing, process visualization (Supervisory Control And Data Acquisition, or SCADA), data transfer for archival, predictive maintenance or Internet of Things (IoT) networks have different requirements, but a common one is the need for transferring large amounts of information. A decisive factor during the planning and realization phase is the knowledge of the protocol and system limitations and performance numbers when a large data volume is involved. Although tools or mechanisms [1] already exist to provide some information, and performance tests have been made ([2] Chapter 13 "Performance", or [3]), in order to systematically test a possible OPC-UA system and its performance measures and to provide decisive information to support an OPC-UA system choice based on custom and mixed criteria, a test platform for the benchmarking of OPC-UA systems has been developed.

This paper contains five additional sections. The first one, Section II, presents the requirements of the developed test platform. Section III describes the parameters under consideration for the study, while the test procedure itself and the details about the considered use-cases and information of interest are discussed in Section IV. Section V describes the setup for the tests and in Section VI some preliminary results are presented, as well as an outlook for future work on this matter.

II. TEST PLATFORM

In order to gather data in a consistent and reproducible manner, a test program was developed to permit the execution of test procedures in a flexible way. The technical goals set for the implementation were:

a) The test procedure should not be hardcoded in the program, in order to allow quick modifications of the test procedures for fast parameter fine-tuning.

b) Test variations should be easy to describe, so that experiments based on previous tests are easier to create and execute.

c) It should be possible to store, reproduce and execute different test procedures.

```
### TEST
test: Concurrent requests
name: concurrent_requests
cycles: 100

concurrency: 1
resource: ns=4;s=MAIN.valueDesc
measure

cycles: 25

count: 1
concurrency: 40
measure

count: 40
measure
```

Figure 1. Sample test description document.

 ISBN: 978-1-61208-437-4

d) Test execution times should be stored in a machine-readable format in order to allow further data processing like statistic generation.

In order to enable the execution of tests in a flexible way, a test description document was created. One or multiple tests can be described in a single file, and measurements can be issued with a simple command. A sample test described on this format is shown in Figure 1. The commands used will be described on the coming sections.

III. PARAMETERS UNDER TEST

For the results discussed in this paper, the following parameters were considered, and thus are available in the test description document for adjustment:

a) Node. The OPC-UA node under test can be specified with this parameter, effectively selecting the amount of data that will transferred per node.

b) Number of cycles. For the purpose of calculating average execution times, the total number of execution cycles can be defined per measurement, and the measurements can be averaged per cycle (combinable with concurrency and number of nodes per request).

c) Concurrency. This parameter defines the number of requests that are sent at the same time to the server, combinable with number of cycles and number of nodes per request. A pool of threads is initialized, and all request workers are instantiated to fetch the defined node at the same time. The test execution waits for all threads to be completed; therefore, the measurements derived from using the concurrency parameter include the amount of time taken by the longest request thread.

d) Number of nodes per request. As described in [2], p. 125, OPC-UA requests can contain a list of nodes to read or write in order to reduce overhead. As such, this parameter determines the number of nodes fetched per read/write request. If this parameter is set to any value greater than 1, the same node is requested multiple times. The number of nodes per request is combinable with the number of cycles and concurrency parameters.

e) Security. Determines the preference for a secured, encrypted endpoint when connecting to an OPC-UA server. OPC-UA endpoints are sorted in descending order of security, and when security is set to *on*, the first endpoint from the available list is chosen, otherwise the last one when set to *off*.

IV. TEST PROCEDURE AND DESCRIPTION OF GATHERED DATA

In order to draw representative conclusions with the different tests, four variables were made available on all PLC programs with varying sizes:

1) Real variable with a total memory usage of 32 bits (REAL), representing a typical single numerical variable.
2) Array of 512 integer numbers of 16 bit each totaling 1 kilobyte of payload (PAGE), representing an average visualization page containing multiple variables.
3) Byte array of 20.000 elements (SDD), representing a sample self-description document in text format.
4) Byte array of 65.535 elements (IMAGE), representing a small image in binary format.

With the aforementioned resources at the disposal on the PLCs, the following tests were conducted:

a) Data volume. The amount of time required to fetch nodes containing variables or different data sizes was tested. Each of the 4 variables was requested in blocks of 1 or 40, one thousand times (one thousand requests of one node/40 nodes each).
b) Grouped requests. The benefit and reduction of overhead by fetching multiple nodes in one request was tested. The 4 variables were fetched in grouped requests of 40 and 400 variables per request and run 100 and 50 cycles respectively, and the times were, after calculating the average time per variable fetched, compared with single variable requests.
c) Concurrent requests. The channel and server efficiency responding to multiple concurrent requests were tested by sending multiple requests at the same time for fetching individual nodes.
d) Security. The impact of the transport encryption was the main purpose of this test. The four variables under test were fetched thousand times in a single variable per request basis, once through a secure endpoint, once through an unsecured endpoint.

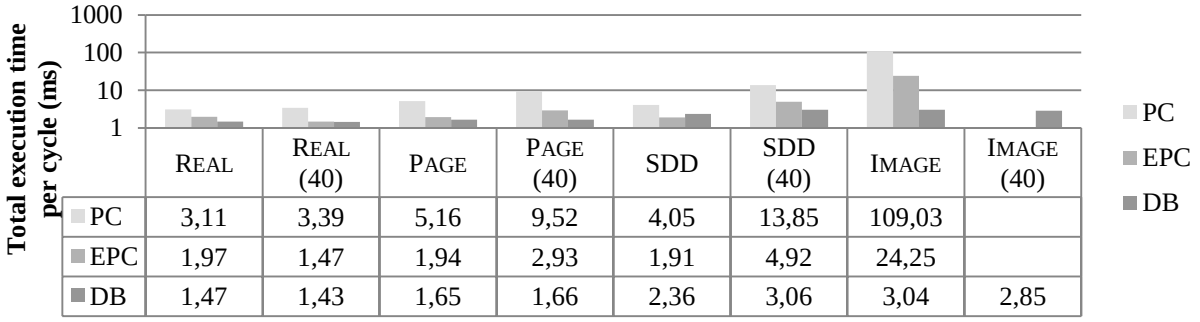
The test description document is interpreted by the test platform program line by line, creating new test instances as required and configuring the test instance appropriately. Upon reading a *measure* command, the current test instance is executed and the execution times and other data are gathered in a file. The following data is contained in the produced file as the result for the test, in addition to the test name and timestamp:

a) Number of resources n_r fetched in total during the test, calculated as $n_r = n_c \cdot n_t$ where n_c is the number of nodes per request, and n_t is the number of threads spawned at the same time.
b) Total duration t taken for the complete test instance to execute.
c) Average time per cycle $\bar{t_c}$ calculated as $\bar{t_c} = t/c$ where c is the number of execution cycles as defined in the test description document (or 1 by default).
d) Average time per resource $\bar{t_r}$ calculated as $\bar{t_r} = \bar{t_c}/n_r$.
e) Standard deviation σ calculated as $\sigma = \left(\sum_{c=0}^{n_c}(t_c - \bar{t_c})^2\right)/n_c$.
f) Coefficient of variation c_v calculated as $c_v = \sigma/\bar{t_c}$.

V. EQUIPMENT UNDER TEST AND SETUP

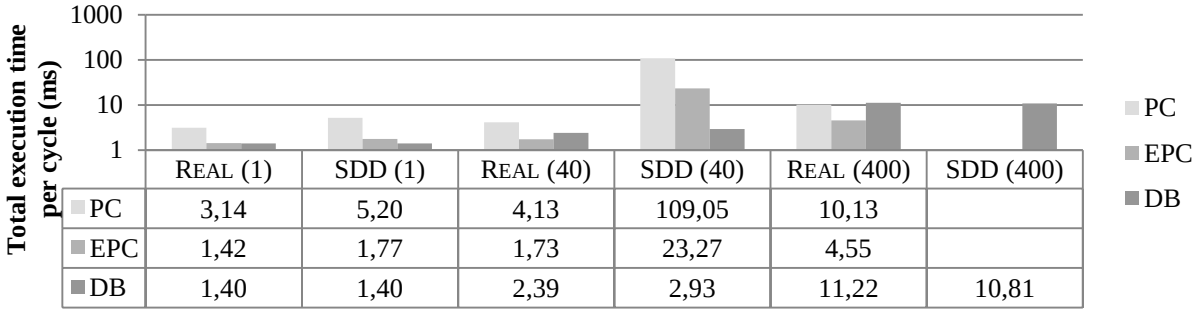
The aforementioned test procedure was executed against three PC-based PLCs running compatible OPC-UA

interfaces: a high-end Microsoft Windows 7-based PLC (PC) running on a dual core processor and using traditional hard disk drives for storage, a lower-end Microsoft Windows Embedded-based PLC (EPC) running on a single core processor and using a flash-based memory storage device, and a development board running an embedded OPC-UA implementation. The client test platform was executed in a standard PC computer, running Microsoft Windows 7 and a .NET OPC-UA stack implementation, and connected directly



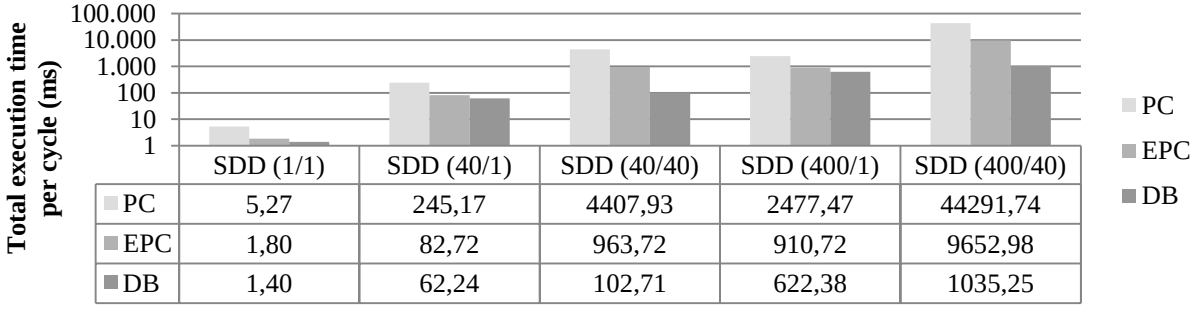
	REAL	REAL (40)	PAGE	PAGE (40)	SDD	SDD (40)	IMAGE	IMAGE (40)
PC	3,11	3,39	5,16	9,52	4,05	13,85	109,03	
EPC	1,97	1,47	1,94	2,93	1,91	4,92	24,25	
DB	1,47	1,43	1,65	1,66	2,36	3,06	3,04	2,85

a) Data volume test. Total execution times for the four different variables by fetching 1 and 40 nodes per request, respectively.



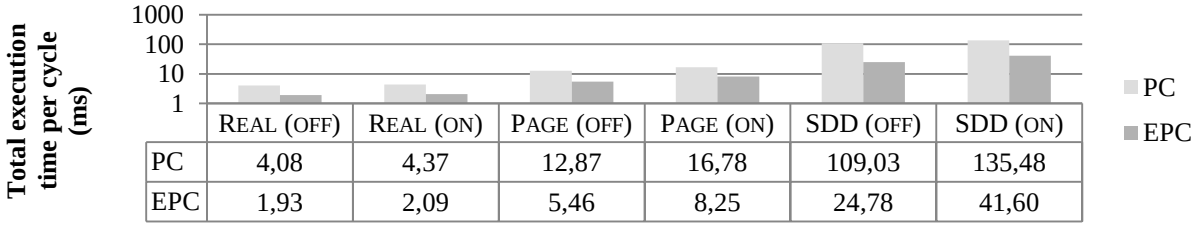
	REAL (1)	SDD (1)	REAL (40)	SDD (40)	REAL (400)	SDD (400)
PC	3,14	5,20	4,13	109,05	10,13	
EPC	1,42	1,77	1,73	23,27	4,55	
DB	1,40	1,40	2,39	2,93	11,22	10,81

b) Grouped requests test. Total execution times for the REAL and SDD variables by fetching 1, 40 and 400 nodes per request, respectively.



	SDD (1/1)	SDD (40/1)	SDD (40/40)	SDD (400/1)	SDD (400/40)
PC	5,27	245,17	4407,93	2477,47	44291,74
EPC	1,80	82,72	963,72	910,72	9652,98
DB	1,40	62,24	102,71	622,38	1035,25

c) Concurrent requests test. Total execution times for the SDD variable by spawning 1, 40 and 400 parallel threads, with 1 and 40 nodes per request (n_t/n_c).



	REAL (OFF)	REAL (ON)	PAGE (OFF)	PAGE (ON)	SDD (OFF)	SDD (ON)
PC	4,08	4,37	12,87	16,78	109,03	135,48
EPC	1,93	2,09	5,46	8,25	24,78	41,60

d) Security test. Total execution times for the Real, Page and SDD variables through an unsecured and an encripted channel, respectively.

Figure 2. Comparison of execution times of the different tests cases.

and separately for each test with every equipment under test with a Cat 6 network cable working in a 100 Mbit/s operation mode. Since in most of the cases the OPC-UA server implementation is not an interchangeable component in the PLC, the complete system was tested, measuring the total time taken by a request to be serialized and processed by the OPC-UA client (t_0), the request to travel to the OPC-UA server (t_1), the request to be deserialized and processed by the OPC-UA server (t_2), the variable to be fetched internally on the PLC (t_3), the variable to be processed internally on the PLC (t_4), the variable to be returned to the OPC-UA server (t_5), the response to be processed and serialized by the OPC-UA server (t_6), the response to travel back to the OPC-UA client (t_7) and the response to be deserialized and processed by the OPC-UA client (t_8), as depicted in Figure 3. The test cases described in Section IV were executed on the three PLCs, yielding the execution times depicted in Figure 2.

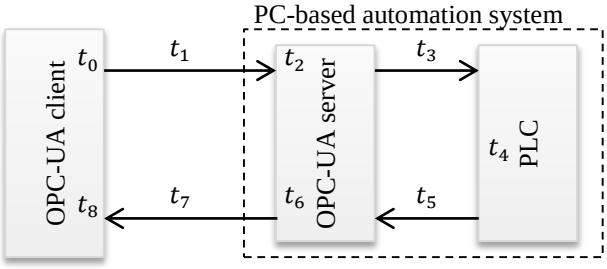
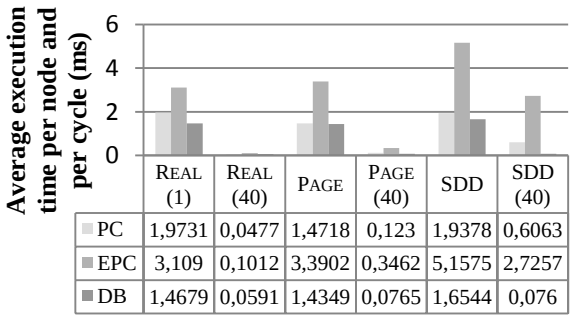


Figure 3. System under test and different processing and travel times.



	REAL (1)	REAL (40)	PAGE	PAGE (40)	SDD	SDD (40)
PC	1,9731	0,0477	1,4718	0,123	1,9378	0,6063
EPC	3,109	0,1012	3,3902	0,3462	5,1575	2,7257
DB	1,4679	0,0591	1,4349	0,0765	1,6544	0,076

Figure 4. Comparison of average execution times (of 1000 execution cycles) per node when fetching the REAL, PAGE and SDD variables in single requests versus 40 nodes per request.

VI. CONCLUSION AND FUTURE WORK

From the gathered preliminary results, some conclusions can be drawn:

a) As expected, requests with single nodes take much overhead, and grouping many variables into a request saves time on all tested implementations, as shown in Figure 4.

b) Given the low RAM available on the EPC and the comparatively slower flash storage used for the virtual memory, larger variables take a considerable time impact, when memory swapping is required and slows down the entire system.

c) Sending multiple requests in parallel is generally a good idea if a large amount of nodes need to be requested at a given time, as all implementations showed drastic time savings by fetching multiple variables separated into requests sent at the same time. Combining multiple threads with multiple nodes per request further improve time savings.

d) As expected, choosing a secure and encrypted endpoint has a speed penalty, although it might not be a deciding factor if security is a requirement. In the tests, a Basic128Rsa15 security policy and a security mode of Sign & Encrypt was used as required by two of the three OPC-UA implementations that implemented secured endpoints.

e) Since the platform tested the whole system, the development board outperformed the other two higher-end systems as the internal communication allowed for faster data retrieval.

In this paper, an OPC-UA benchmarking tool has been presented, in addition to some preliminary results. The testing platform enables the capture of performance information in a flexible and reproducible way, and makes it possible to describe tests using any combination of parameters permitting the compilation of exactly the required data to make decisions about the amount of information that it is possible to transfer in a given application, or the speed limitations when planning a facility. The results presented here are preliminary, and more statistics and parameters are foreseen to be included in the testing application, configurations to better simulate the conditions that a facility will face, as well as a more detailed test scenario including specific PLCs with varied configurations and optimizations. Comparisons between software- and hardware-based PLCs, as well as with variations in the network equipment (like cable length, class and age) and bus couplers are planned to be the subject of further evaluation.

ACKNOWLEDGMENT

F.G.V. would like to thank Pablo Mayer and Fabian Böttinger for their help building and configuring the equipment under test described on this paper, as well as for their invaluable advice on this project.

REFERENCES

[1] "Resource Efficiency Testing, Step by Step Instructions (whitepaper)," 16 May 2014. [Online]. Available: https://opcfoundation.org/wp-content/uploads/2014/05/Certification_Resource_Efficiency_Testing.pdf. [Accessed 22 May 2015].

[2] W. Mahnke, S.-H. Leitner, and M. Damm, OPC Unified Architecture, Springer-Verlag Berlin Heidelberg, 2009.

[3] C. Salvatore and G. Cutuli, "Performance evaluation of OPC UA," in 2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA), Bilbao, 2010, pp. 1-8.

[4] J. Lange and F. Iwanitz, OPC, From Data Access to Unified Architecture, VDE Verlag, 2010.

 ISBN: 978-1-61208-437-4